SCHOOL OF SCIENCE AND TECHNOLOGY, SINGAPORE

SECONDARY 4 PRELIMINARY EXAMINATION

COMPUTING
Paper 2 **7155/02**
Practical (Lab-based)

24 August 2021 (Tuesday) **2 hours 30 minutes**

CANDIDATE NAME			
CLASS		INDEX NUMBER	

Additional Materials: Electronic version of `CREDITCARD.xlsx` data file
Electronic version of `TEAORDER.py` file
Electronic version of `MORETEA.py` file
Insert Quick Reference Glossary

READ THESE INSTRUCTIONS FIRST
Answer **all** questions.

All tasks must be done in the computer laboratory. You are not allowed to bring in or take out any pieces of work or materials on paper or electronic media or in any other form.

Programs are to be written in Python.
Save your work using the file name given in the question as and when necessary.

The number of marks is given in brackets [ ] at the end of each question or part question. The total number of marks for this paper is **50**.

For Examiner's Use	
Task 1	
Task 2	
Task 3	
Task 4	
Total	**50**

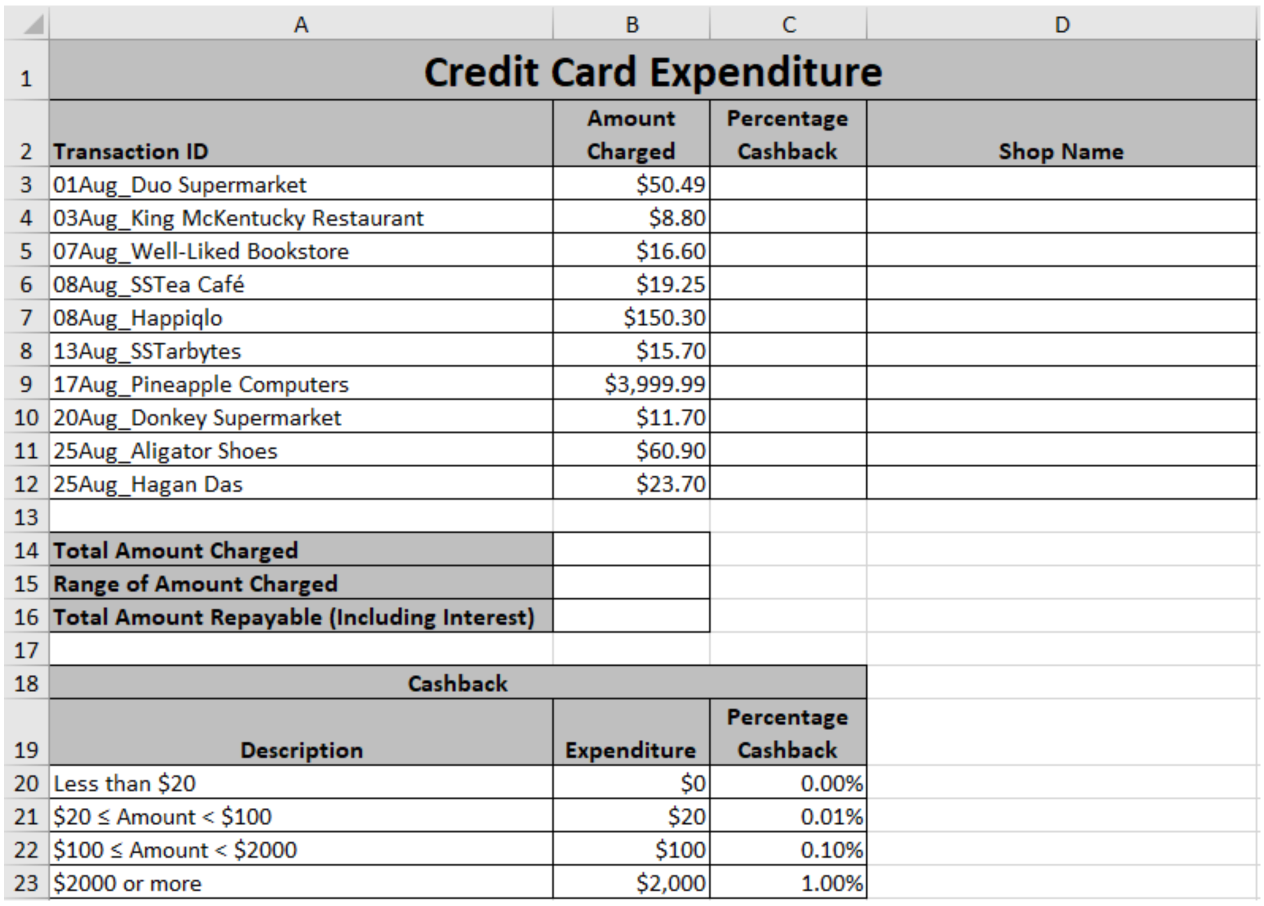
This document consists of **9** printed pages, including this Cover Page.

Task 1

A customer of a credit card company uses spreadsheet software to keep track of his expenditure with his credit card. He lists his spending according to the Transaction ID.

You are required to help this customer finish up the spreadsheet.

Open the file **CREDITCARD.xlsx** and you will see the following data.



	A	B	C	D
1	**Credit Card Expenditure**			
2	**Transaction ID**	**Amount Charged**	**Percentage Cashback**	**Shop Name**
3	01Aug_Duo Supermarket	$50.49		
4	03Aug_King McKentucky Restaurant	$8.80		
5	07Aug_Well-Liked Bookstore	$16.60		
6	08Aug_SSTea Café	$19.25		
7	08Aug_Happiqlo	$150.30		
8	13Aug_SSTarbytes	$15.70		
9	17Aug_Pineapple Computers	$3,999.99		
10	20Aug_Donkey Supermarket	$11.70		
11	25Aug_Aligator Shoes	$60.90		
12	25Aug_Hagan Das	$23.70		
13				
14	**Total Amount Charged**			
15	**Range of Amount Charged**			
16	**Total Amount Repayable (Including Interest)**			
17				
18	**Cashback**			
19	**Description**	**Expenditure**	**Percentage Cashback**	
20	Less than $20	$0	0.00%	
21	$20 ≤ Amount < $100	$20	0.01%	
22	$100 ≤ Amount < $2000	$100	0.10%	
23	$2000 or more	$2,000	1.00%	

Save your file as **MYCREDITCARD_**<your class>_<index number>_<your name>**.xlsx**

1 In cell **B14** enter a formula that uses (an) appropriate function(s) to find the total amount charged. [1]

2 In cell **B15** enter a formula that uses (an) appropriate function(s) to find the range of the amount charged for the transactions. [1]

3 The credit card company offers a varying percentage of cashback on expenditure, depending on the amount spent in each transaction.

In cells **C3** to **C12** enter a formula that uses (an) appropriate function(s) to search for the **Percentage Cashback** in the **Cashback** table. [2]

4 The **Transaction ID** of each transaction consists of the date of the transaction followed by the shop name.

In cells **D3** to **D12** enter a formula that uses (an) appropriate function(s) to obtain the **Shop Name** from the Transaction ID. [2]

5 In cells **A3** to **A12**, use a formatting tool to change the colour of the cell to red if it corresponds to an expenditure of more than $50 as reflected in cells **B3** to **B12**. [2]

6 The credit card company also allows their customers to pay their credit card bill in 6 equal monthly payments, with 12% of interest per annum, compounded monthly.

In cell **B16** enter a formula that uses (an) appropriate function(s) to find the total amount repayable, including interest, if they take this option. [2]

Save and close your file.

Task 2 begins on the next page.

Task 2

The following program is a point-of-sale (POS) system which reads in a customer's drink order and displays the total bill.

```
menu = ['OOLONG MILK TEA', 'EARL GREY MILK TEA', 'MATCHA LATTE']
price = [4.9, 4.5, 6.9]
print('Welcome to SSTea Inc!')
order = input('What tea would you like to order? ')
bill = price[menu.index(order)]
print('Your total bill is ${}.'.format(bill))
print('Thank you. Please come again!')
```

Open the file **TEAORDER.py**

Save the file as **MYTEAORDER**_<your class>_<index number>_<your name>**.py**

7 Edit the program to add "YEO'S CHRYSANTHEMUM TEA" to the menu at the price of $1.90 per order. [1]

Save your program.

8 Edit the program to accept the order input in any letter case. [1]

Save your program.

9 Edit the program to only accept orders that are on the menu. A suitable error message must be displayed if the order input is not in the menu. The program must loop until a valid order is input. [3]

Save your program.

10 Edit the program to accept more than one order. The program must ask the user how many drinks are to be ordered. Validation is not required for this input. After the addition of each order, the program must display the subtotal of the customer's bill. After all the orders have been made, the program must display the total bill. [3]

Save your program.

11 Edit the program to apply a 20% discount on the final total bill if the customer orders more than 3 drinks. [2]

Save your program.

Task 3

To cater for more customer customisations, SSTea Inc. decides to add a new function to their existing POS program with the following options:

- **Size:** Small (+$0), Medium (+$1), Large (+$1.50)
- **Sugar level:** 0%, 25%, 50%, 75%, 100%
- **Add topping:** Pearls (+$0.50), Jelly (+$1.00)

The function allows a user to customise the size and sugar level of their drink, and choose toppings to add. If the size or sugar level input is incorrect, the function will void all selections. Otherwise, the function will return the new cost of the drink.

There are several syntax errors and logic errors in the function.

```
class drinkOption(base_cost) :
    add_cost = base_cost
    reject = True
    print('Small +$0, Medium +$1.00, Large +$1.50")
    size = input("Upsize? [S, M, L] : ").upper()
    sugar_level = input("Sugar level [0, 25, 50, 75, 100] : ")
    pearls = input("Add Pearls +$0.50  [Y]es [N]o : ").upper()
    jelly = input("Add Jelly +$1.00 [Y]es [N]o : ").upper()
    if size == "m":
        add_cost += 1.0
    elif size == "L" :
        add_cost += 1.5
    elif size == "S" :
        add_cost = 0
    elif:
        reject = True
    if sugar_level % 25 != 0:
        reject = True
    if pearls == "Y":
        add_cost += 0.55
    elif jelly == "Y":
        add_cost += 1.0
    if reject:
        return base_cost
        return add_cost

menu = ["OOLONG MILK TEA", "EARL GREY MILK TEA", "MATCHA LATTE"]
price = [4.9, 4.5, 6.9]
print("Welcome to SSTea Inc!")
order = input("What tea would you like to order? ")
bill = price[menu.index(order)]
bill = drinkOption(bill)
print("Your total bill is ${}.".format(bill))
print("Thank you. Please come again!")
```

Open the file **MORETEA.py**

[Turn over

Save the file as **MYMORETEA_**<your class>_<index number>_<your name>**.py**

12 Identify **and** correct the errors in the program so that it works according to the requirements given. [10]

Save your program.

Task 4 begins on the next page.

Task 4

A school is organising an upcoming inter-house competition named Three Dragon Ante.
There are a total of five Houses: Blue, Green, Purple, Red, Yellow. Students can form their own teams among the same House members. Team names consists of the House initials followed by a two-digit number. Each team name is unique.

The competition is played over many matches. Each match involves the participation of three teams. Two teams of the same House may participate in the same match. A team may participate in more than one match.

Each match can produce one of the following results:

1. One winning team, two losing teams.
2. Two winning teams, one losing team.

Each winning team contributes 1 point to the House it represents.

The school wants a program that will tabulate the results and display the relevant statistics with the following criteria:

- Take as input a match report: a comma separated string, "`N,A,B,C`", where *N* denotes the number of winning teams (either 1 or 2), and *A*, *B*, *C* are team names. If *N* = 1, *A* is the winning team; if *N* = 2, *A* and *B* are both winning teams.
- Display a suitable message each time one or more of the team names in an input is not of the correct format. It is not necessary to check the number of winning teams indicated or the number of team names inputted.
- Loop until "`end`" is inputted to indicate that no more match reports need to be entered.
- Display the total points accumulated by each House on separate lines, in the following order: Blue, Green, Purple, Red, Yellow.

A suitable message must be used for each input and output.

13 Write a program for the given criteria. [11]

Save your program as **MATCHREPORT**_<your class>_<index number>_<your name>**.py**

[Turn over

14 When your program is working, use the following test data to show your test result.

```
1,R04,P01,B03
1,B03,P01,Y02
2,Y02,G05,P06
1,R04,Y02,B03
1,P01,Y02,P06
end
```

Take a screenshot of your result and save it as:

MATCHREPORTTEST_<your class>_<index number>_<your name> [2]

Save your file in either **.jpg** or **.png** format.

15 Save your program as **MATCHREPORT2**_<your class>_<index number>_<your name>**.py**

Extend your program with the following criteria:

- Display the total number of teams that participated in the competition.
- Display each House name, followed by all the teams that represented the House.

Appropriate output formatting must be used for each output. [4]

Save your program.

16 Save your program as **MATCHREPORT3**_<your class>_<index number>_<your name>**.py**

The school wants to give out a trophy to the best team. It is deemed that a result of a match of "1,*A*,*B*,*C*" implies that *A* is better than *B* and *C*. Similarly a result "2,*A*,*B*,*C*" implies that *A* and *B* is better than *C*.

Write a function, ***FindOrdering()***, which returns a possible ordering of the teams such that a better team always comes before its opponent. If no such ordering exists, the function should return `0`.

For example, for the first line of the test case in Question 14, a possible ordering would be `R04 P01 B03`, with the two losing team being in any order. The second line of input shows that B03 won against P01 and thus B03 must be placed before P01 in the output. Therefore, the updated possible order would be `R04 B03 P01 Y02`.

Hence, a possible ordering for the entire test case is: `R04 G05 B03 P01 Y02 P06`

However, no such ordering exists for the following test data, and so a `0` should be returned:

```
1,P01,Y02,R04
2,Y02,R04,P01
```

[3]

Save your program.

--- END OF PAPER ---